Unification and Resolution (Continued)

Friday, 01 October 2004

William H. Hsu
Department of Computing and Information Sciences, KSU
http://www.kddresearch.org
http://www.cis.ksu.edu/~bhsu

Reading:
Chapter 10, Russell and Norvig
(for PS3 Read & Explain Pairs, due Friday 08 Oct 2004)
Lecture Outline

• Today’s Reading
  – Chapter 9, Russell and Norvig
  – Recommended references: Nilsson and Genesereth (excerpt of Chapter 5 online)

• Last Week’s Reading: Chapter 9, R&N

• Previously: Propositional and First-Order Logic
  – Two weeks ago
    • Logical agents: KR, inference, problem solving
    • Propositional logic: normal forms, sequent rules
    • Predicates and terms
    • First-order logic (FOL): quantifiers
  – Last week
    • FOL agents; frame problem; situation calculus, successor-state axioms
    • FOL KBs and forward search using sequent rules (sound but incomplete set)

• Today: Backward Inference
  – Resolution refutation (sound and complete proof procedure)
  – Computability (decidability) issues
Example [1]
English to FOL to CNF (Clausal Form)
Offline Exercise: Read-and-Explain Pairs

- For Class Participation (PS3, MP4)
- With Your Term Project Partner or Assigned Partner(s)

- Read: Chapter 9 (esp. 9.2, 9.5), Chapter 10 R&N 2e
- By Fri 08 Oct 2004, Fri 15 Oct 2004
Depth-first search from a start state X:

dfs(X) :- goal(X).
dfs(X) :- successor(X,S), dfs(S).

No need to loop over S: successor succeeds for each

Appending two lists to produce a third:

append([],Y,Y).
append([X|L],Y,[X|Z]) :- append(L,Y,Z).

query: append(A,B,[1,2]) ?
answers: A=[] B=[1,2]
         A=[1,2] B=[]
Completeness of Resolution

- Any Set of Sentences $S$ Is Representable in Clausal Form (Last Class)
- Assume $S$ Is Unsatisfiable, and in Clausual Form
- (By Herbrand’s Theorem) Some Set $S'$ of Ground Instances is Unsatisfiable
- (By Ground Resolution Theorem) Resolution Derives $\bot$ From $S'$
- (By Lifting Lemma) $\exists$ A Resolution Proof $S \mathcal{O}_n \bot$

Figure 9.13 p. 301 R&N 2e
Decidability Revisited

• See: Section 9.7 Sidebar, p. 288 R&N

• Duals (Why?)

\[
\frac{L_{VALID}}{L_{SAT}} \quad \frac{L_{VALID}}{L_{SAT}}
\]

• Complexity Classes

• Understand: Reduction to \( L_d, L_H \)
A substitution $\sigma$ unifies atomic sentences $p$ and $q$ if $p\sigma = q\sigma$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Knows}(John, x)$</td>
<td>$\text{Knows}(John, Jane)$</td>
<td>${x/Jane}$</td>
</tr>
<tr>
<td>$\text{Knows}(John, x)$</td>
<td>$\text{Knows}(y, OJ)$</td>
<td>${x/John, y/OJ}$</td>
</tr>
<tr>
<td>$\text{Knows}(John, x)$</td>
<td>$\text{Knows}(y, \text{Mother}(y))$</td>
<td>${y/John, x/\text{Mother}(John)}$</td>
</tr>
</tbody>
</table>

**Idea:** Unify rule premises with known facts, apply unifier to conclusion
E.g., if we know $q$ and $\text{Knows}(John, x) \Rightarrow \text{Likes}(John, x)$
then we conclude $\text{Likes}(John, Jane)$
$\text{Likes}(John, OJ)$
$\text{Likes}(John, \text{Mother}(John))$

- **Most General Unifier** (Least-Commitment Substitution)
- **See:** Examples (p. 271 R&N, Nilsson and Genesereth)

*Adapted from slides by S. Russell, UC Berkeley*
Example [2]
Unification and Resolution
Logic Programming – Tricks of The Trade [1]: Dealing with Equality

• Problem
  – How to find appropriate inference rules for sentences with =?
  – Unification OK without it, but…
  – $A = B$ doesn’t force $P(A)$ and $P(B)$ to unify

• Solutions
  – **Demodulation**
    • *Generate substitution from equality term*
    • Additional sequent rule: p. 284 R&N
  – **Paramodulation**
    • More powerful
    • *Generate substitution from WFF containing equality constraint*
    • e.g., $(x = y) \lor P(x)$
    • Sequent rule sketch: p. 284 R&N
Example [3]
Demodulation and Paramodulation
Logic Programming – Tricks of The Trade [2]: Resolution Strategies

- **Unit Preference**
  - Idea: Prefer inferences that produce shorter sentences (compare: Occam’s Razor)
  - How? Prefer *unit clause* (single-literal) resolvents
  - Reason: trying to produce a short sentence ($\bot \equiv \text{True} \Rightarrow \text{False}$)

- **Set of Support**
  - Idea: try to eliminate some potential resolutions (prevention as opposed to cure)
  - How? Maintain set SoS of resolution results and always take *one resolvent* from it
  - Caveat: need right choice for SoS to ensure completeness

- **Input Resolution and Linear Resolution**
  - Idea: “diagonal” proof (proof “list” instead of proof tree)
  - How? Every resolution combines some input sentence with some other sentence
  - Input sentence: *in original KB or query*
  - Generalize to *linear resolution*: include any ancestor *in proof tree* to be used

- **Subsumption**
  - Idea: eliminate sentences that sentences that are more specific than others
  - E.g., $P(x)$ *subsumes* $P(A)$
Logic Programming – Tricks of The Trade [3]:
Indexing Strategies

• **Store and Fetch**
  – Idea: store knowledge base in *list* of conjuncts
  – **STORE**: constant, i.e., $O(1)$ worst-case running time
  – **FETCH**: linear, i.e., $O(n)$ time

• **Table Based**
  – Idea: store KB in *hash table* (key: ground literals)
  – **STORE**: $O(1)$
  – **FETCH**: $O(1)$ expected case
  – Problems
    • Complex WFFs (other than negated atoms)
    • Variables
      – Solution: *implicative normal form matching* (Figure 10.1, p. 301 R&N)

• **Tree-Based**
  – What if there are many clauses for a predicate? (e.g., *Brother* (012-34-5678, $x$))
  – Type of combined indexing: *joint primary key* – predicate and argument symbols
  – May need background knowledge for semantic query optimization (SQO)
Logic Programming – Tricks of The Trade [4]: Compilation

• Intermediate Languages
  – Abstract machines
    • Warren Abstract Machine (WAM)
    • Java Virtual Machine (JVM)
  – Imperative intermediate representations (IRs)
    • C/C++
    • LISP / Scheme / SML – functional languages with imperative features

• Use in Genetic Programming (GLP): Later
• Beyond Scope of CIS 730: Compiling with Continuations (Appel)

Adapted from slides by S. Russell, UC Berkeley
Summary Points

- Previously: FOL, Forward and Backward Chaining, Resolution
- Today: More Resolution Theorem Proving, Prolog, and Unification
  - Review: resolution inference rule
    - Single-resolvent form
    - General form
  - Application to logic programming
  - Review: decidability properties
    - FOL-SAT
    - FOL-NOT-SAT (language of unsatisfiable sentences; complement of FOL-SAT)
    - FOL-VALID
    - FOL-NOT-VALID
  - Unification
- Next Week
  - Intro to classical planning
  - Inference as basis of planning
Terminology

- Properties of Knowledge Bases (KBs)
  - Satisfiability and validity
  - Entailment and provability
- Properties of Proof Systems
  - Soundness and completeness
  - Decidability, semi-decidability, undecidability
- Resolution
- Refutation
- Satisfiability, Validity
- Unification
  - Occurs check
  - Most General Unifier
- Prolog: Tricks of The Trade
  - Demodulation, paramodulation
  - Unit resolution, set of support, input / linear resolution, subsumption
  - Indexing (table-based, tree-based)